TP n° 9 – Satisfiabilité

etienne.jl.simon@gmail.com thomas.h@ens.fr
thomash.fr/teaching/

Dans ce sujet nous allons nous intéresser au problème de satisfiabilité d'une formule logique. Une formule logique est représentée par le type `char formula` et elle est dite satisfiable s'il existe des valeurs de vérité pour les variables qui rendent la formule vraie.

```
type 'a formula =
| Var of 'a
| Not of 'a formula
| And of 'a formula * 'a formula
| Or of 'a formula * 'a formula
```

Question 1. Écrire la fonction `eval : ('a -> bool) -> 'a formula -> bool` qui étant donné les valeurs de vérité des variables renvoie la valeur de vérité de la formule.

1 CNF

Afin de faciliter les manipulations de formules logiques, on préfère les avoir sous forme normale conjonctive (Conjunctive Normal Form), c'est à dire comme une conjonction (ET) de disjonctions (OU) de littéraux. Un littéral étant une variable logique ou sa négation.

```
type 'a litteral =
| Pos of 'a
| Neg of 'a
```

Question 2. Écrire la fonction `cnf : 'a formula -> 'a litteral list list` qui étant donné une formule renvoie sa forme normale conjonctive.

Question 3. Lorsqu'une formule est sous forme normale conjonctive, on peut la simplifier en enlevant les disjonctions tautologiques (qui contiennent à la fois une variable et sa négation) et en propageant les valeurs de vérités forcées (une variable a une valeur de vérité forcée si elle est seule dans une disjonction).

Écrire la fonction `simplify : 'a litteral list list -> 'a litteral list list` qui simplifie une formule en CNF.

2 2-SAT

Le problème de satisfiabilité est NP-complet, cependant si toutes les disjonctions contiennent au plus 2 littéraux, il existe un algorithme polynomial pour tester la satisfiabilité d'une formule.

Question 4. Observer qu'une disjonction de deux littéraux peut être réécrite comme une implication. En déduire un algorithme polynomial pour résoudre 2-SAT.

Question 5. Écrire la fonction `two_sat : 'a litteral list list -> ('a * bool) list option` qui étant donné une formule en CNF renvoie `None` si elle n'est pas satisfiable et si elle est satisfiable `Some values` où `values` est la liste des valeurs de vérité qui rendent la formule vraie.

3 SAT

Le problème de satisfiablité étant NP-complet, nous allons le résoudre en testant les différentes valeurs de vérité possible et en revenant en arrière en cas de blocage.

Question 6. Bien que cet algorithme soit exponentiel dans le pire des cas, il peut se révéler très rapide en pratique si on choisit bien les variables sur lesquelles on fait nos hypothèses.

Proposez des heuristiques pour le choix des hypothèses à tester.

Question 7. Écrire la fonction `sat : 'a litteral list list -> ('a * bool) list option` qui résout le problème de satisfiabilité général.

4 SMT

L'une des applications du problème SAT est la preuve automatique de propositions mathématiques. Cependant, les propositions mathématiques contiennent des expressions arithmétiques et booléennes à la place des variables logiques.

On se limite ici aux expressions linéaires à deux variables (A et B) représentées par le type suivant.

```
type arith_expr =
| Const of int
| A
| B
| Add of arith_expr * arith_expr
| Sub of arith_expr * arith_expr

type bool_expr = Greater of arith_expr * arith_expr
```

Question 8. Écrire une fonction `linear2 : bool_expr litteral list -> bool` qui étant donné une liste d'expressions booléennes vérifie qu'il existe des valeurs pour les variables A et B qui satisfont toutes les expressions de la liste.

La méthode de résolution des problèmes SMT (Satisfiability Modulo Theories) consiste à remplacer chaque expression booléenne par une variable logique et à passer la formule obtenue au solveur SAT. Si la formule n'est pas satisfiable alors *a fortiori* la proposition de départ ne peut pas être vraie. Si le solveur SAT propose des valeurs de vérité qui satisfont la formule, il faut encore vérifier que ces valeurs de vérité sont compatibles avec la théorie. Par exemple la formule associée à $(A > 3 \vee A > 2) \wedge (0 > A \vee 1 > A)$ est satisfiable alors que cette proposition est toujours fausse. On utilise donc le solveur pour la théorie (ici `linear2`) pour vérifier que la solution proposée est bien correcte. Si ce n'est pas le cas, on ajoute dans la formule logique le fait que ces valeurs de vérité sont interdites et on recommence.

Question 9. Écrire la fonction `smt : bool_expr formula -> bool` qui dit si une proposition mathématique est satisfiable.